Engineering with Computers (2004) 20: 65–74
DOI 10.1007/s00366-004-0270-x

ORIGINAL ARTICLE

A. Csabai · P. Xirouchakis

On the medial surface approximations of extrusions

Received: 20 February 2001 / Accepted: 12 July 2002 / Published online: 25 February 2004
© Springer-Verlag London Limited 2004

Abstract Generating the medial surface for a general boundary representation model raises several difficulties. Problems might emerge from the complexity of the resulting equations, singularities caused by unforeseen relative boundary element positions and orientations, etc. The majority of the current algorithms are based on the topology of the boundary representation model and produce wireframes composed of straight lines regardless of the real medial surfaces. Many of the solids used in engineering can be represented by extrusions, delimited by a cross-section and an extrusion distance. This paper develops a fast and efficient method for creating the facetted approximations of the medial surfaces of extrusions generated by sweeping along the normal direction to the generating cross-section.

Keywords Disassembly evaluation · Extrusions · Interference analysis · Layout modelling · Medial axis transform · Medial surface

Introduction

The medial surface (MS, or as referenced by other works, medial axis transform, or MAT) was introduced by Blum [1] and is defined as the infinite set of the centres of the maximal inscribed spheres (MIS) of a 3D object. Similarly, in 2D, the MAT is the locus of the centres of the maximal inscribed circles (MICs). The MS technology has many applications in engineering design, for example geometric reasoning [19], collision detection, feature recognition [13], shape interrogation [6], mesh generation [7, 8, 19], etc. However, the number of systems that apply and exploit the advantages of the MS representation is quite small because of the absence of MS calculation methods that work under every circumstance and generate exact or fairly approximate medial surfaces. Several attempts have been made to work out such general methods but each of them has its own drawbacks such as numeric instability or crude approximation. Nevertheless, by considering only a well-defined subset of engineering design problems one can identify subclasses of the occurring geometric forms with properties that can be efficiently exploited during the calculation of medial surfaces.

The main application fields of the 2D MAT are NC machining and FEM mesh generation. In the pocket machining technique introduced by Held et al. [9] a MAT-based proximity map helps to keep track of the distance relations inside the pocket. Gürsoy and Patrikalakis used MAT to subdivide a complex surface and to get the local element size to generate fine triangular meshes within subregions [7, 8, 6]. In the 3D case, the majority of the approaches work with B-rep models. Reddy and Turkiyyah used a constructive method to build up the dual of the medial surface, i.e., a set of connected Delauney tetrahedra [11]. This is also the basis of the work of Renner and Stroud [14, 13], but they added several optimisations to the initial method such as the multiple start point [14] and divide-and-conquer techniques (Stroud, Renner, Xirouchakis, personal correspondence). The work of Reddy-Turkiyyah and Renner-Stroud approximates the MS by a wireframe composed of straight lines. Sherbrooke et al. found an alternative way of handling the problem; they use a differential MS edge traversing approach maintaining the sorted list of distance events [17, 18]. Besides working on B-rep models, other pieces of work addressed the problem of MS generation for CSG models. For example, Dutta and Hoffmann analysed the Voronoi surfaces of simple, but frequently used shape elements in order to calculate the MS of CSG objects [4]. Both techniques have the common problem of robustness due to the high degree systems of equations and numerical instability. Generally, the cost of increasing the robustness [10] has to be paid from the performance of the algorithm.

A. Csabai · P. Xirouchakis (✉)
Laboratory of CAD/CAM,
Swiss Federal Institute of Technology, ME Ecublens,
1015 Lausanne, Switzerland
E-mail: paul.xirouchakis@epfl.ch

In addition to the Renner-Stroud algorithm there are several other algorithms; four examples include those by Reddy and Turkiyyah [11], Sherbrooke, Patrikalakis and Brisson [17], Sheehy, Armstrong and Robinson [15, 16] and Etzion and Rappoport [5]. Direct comparison with these algorithms has not been possible because only the Renner-Stroud code was available for testing.

The first algorithm, by Reddy and Turkiyyah, is similar to that of the first Renner-Stroud algorithm, which was derived from it. However, it suffers from the same drawbacks in that it can find multiple, coincident MAT vertices where more than four elements delimit a vertex. In addition the Renner-Stroud version finds multiple start-points, thus improving the efficiency of the algorithm.

The algorithm by Sherbrooke, Patrikalakis and Brisson uses a method to trace the MAT edges, ''seam edges'' looking for junction points. Once the edges and the vertices have been determined the MAT surfaces are generated. There are some similarities with the algorithm described in this paper. The edges are traced using the centre points of the circles, with the junction points determined using a pattern recognition technique. However, the special case for which the algorithm described here has been developed allows some optimisation which is not possible for the general case.

The Sheehy, Armstrong and Robinson algorithm presents yet another approach, using points distributed on the body to generated Delaunay tetrahedra, the centres of the associated maximal spheres of which lie on the medial axis surface. In contrast to the previous two algorithms this is not geometry dependent, since the points may lie on any surface type. One difference with the algorithm described here is that the maximal spheres are computed directly rather than using the intermediate notion of the Delaunay structures. Another difference is that the boundary elements themselves are used to delimit the spheres rather than points, which is implicitly more efficient.

Etzion and Rappoport describe a medial axis calculation algorithm where the symbolic and geometric parts are calculated separately. The algorithm uses a proximity structure to calculate the symbolic part which then facilitates the computation of the geometric part. This separation is not done in the algorithm described in this paper.

In addition there are other algorithms based on cellular methods which use thinning to arrive at an object skeleton, but these are sufficiently different from the approach described here, and not to be dealt with here.

In order to provide an example for the MS calculation, we present the basic algorithm of Reddy and Turkiyyah [11], as formulated by Renner and Stroud [12]:

1. Make a list of all relevant boundary entities.
2. Determine all starting points, i.e., vertices where at least three boundary entities meet. These first three entities form a ''seed triangle'' which is placed on a list of triangles to be processed.
3. Pick a triangle from the list to be processed. Find all possible fourth entities which together with the three defined by the triangle bound spheres inside the object. If no candidate fourth elements are found, repeat this step.
4. Sort the list of possible entities in order of increasing distance of the corresponding sphere centre from the previous critical point.
5. Discard any candidates where other boundary entities intersect the sphere.
6. If any fourth entities remain, pick the closest one and form a new tetrahedron. If any triangles of the new tetrahedron already exist then merge the existing triangle into the new tetrahedron, if possible. Add any new triangles bounding the tetrahedron to the list of triangles to be processed.
7. Repeat from Step 3.

It can be seen that, for example, in the case when three edges and one vertex boundary elements define an inscribed sphere then the resulting systems of equations might be composed of three quadratic equations which require numeric solution methods. In the case of many boundary elements this method is computationally unstable and expensive.

Geometric representation

Given a simply connected planar polyline with a n unit normal vector of its plane and a W real number, from the point of view of the internal representation, the planar polyline is composed of line segments (edges) and concave vertices. Convex vertices are not represented explicitly since they do not play a significant role in the MS generation method to be discussed. The polyline is swept along the Wn vector creating the extrusional object and considering the plane of the cross-section laid down horizontally, we can distinguish horizontal and vertical elements in that. The locus of convex/concave vertices of the swept cross-section result in convex/concave vertical side edges, called *convex/concave v-edges*. The edges of the extremal cross-section positions form the horizontal top and bottom edges, called *h-edges* (Fig. 1).

If we generate the MS of such an object (with any kind of algorithm), we can then identify specific types of medial surfaces. These can be classified as follows:

- The locus of the maximal inscribed spheres that are constrained by the top and the bottom faces at the same time are the *h-surfaces* (''horizontal surfaces''). These planar surfaces are parallel to the original cross-section.
- The locus of MISs that are constrained by two non-neighbouring vertical elements (side face or concave *v*-edge) are the *v-surfaces* (''vertical surfaces''). *v*-surfaces are perpendicular to the original cross-section.
- The locus of MISs that are constrained by two neighbouring vertical elements are the *w-surfaces* (''wing surfaces''). Depending on whether both of

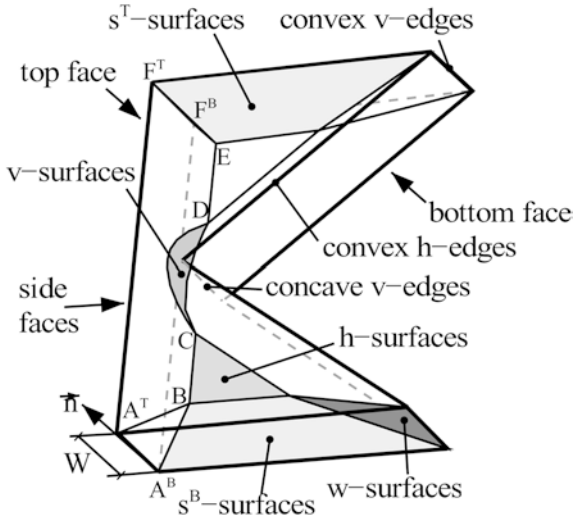

Fig. 1 Entities of the MS representation

them are side faces or either of them is a concave *v*-edge (''vertical edge'') we distinguish *w^F-surfaces* and *w^E-surfaces*, respectively. (Two w^E-surfaces emanate from a concave vertical edge which are perpendicular to each of the incident faces.)

– The locus of MISs that are constrained by a vertical element (a side face or a concave edge) and the bottom or the top face are the *s-surfaces* (''seam-surfaces''). We can distinguish *s^T-surfaces* and *s^B-surfaces* depending on whether one of the constraining elements is the top face or the bottom face, respectively.

Figure 2 illustrates the classification of the mentioned surfaces.

MS generation

MS boundary points

The method starts by finding the points of the boundary curves of the medial surfaces. These points are constrained by at least three elements. The first part of the method works on the planar cross-section since the points of the planar projections of the above curves can be calculated easily in 2D. As the result of the projection side faces become line segments and concave edges

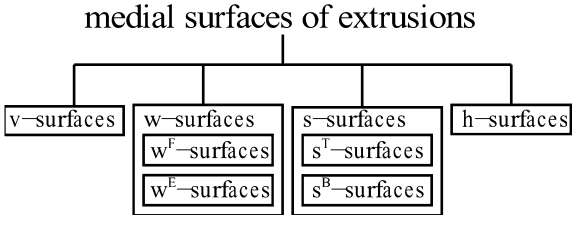

Fig. 2 The classification of extrusional MS surfaces

Table 1 Mapping between the locus of centres of inscribed circles and projections of 3D MS surfaces

Constrained by	Circle diameter	Projection of
Two neighbouring boundary elements	$<W$	w-surface
Two non-neighbouring boundary elements	$<W$	v-surface
Circle diameter	$=W$	h-surface

become concave vertices. The constrained inscribed spheres degenerate into constrained inscribed circles whose dependency on the cross-section element is summarised in Table 1.

The input is a cross section represented by a simply-connected planar polygon. The polygon is represented by a set of boundary elements of two types: edges and concave vertices. In the first step offset circles are introduced in the interior of the cross-section. Each edge has a set of circles touching it arranged equidistantly along it, i.e., each concave vertex has a set of touching circles. Offset circles are swept along edges and around concave vertices so that the currently processed boundary element touches the offset circles. This is done by subdividing the offset lines and arcs based on a predefined preferred distance of the centres of offset circles (a constant). The actual distance is calculated in the following way:

$$a' = \frac{L}{\operatorname{int}\frac{L}{a} + 1}$$

(see Fig. 3). (There is also a minimum number of offset circles per boundary element for the case of too short edges and too blunt concave vertices.) This offsetting is done conforming to a predefined traversal sense which is clockwise with respect to the normal of the cross section. The diameter of the offset circles is equal to the extrusion thickness. These traversed boundary elements become the *reference boundary elements, (RBEs)* of the attached offset circles. The first circles of RBEs and their centre points are called *F-circles* and *F-points*, respectively (Fig. 3).

The next step is to constrain the offset circles by cross-section boundary elements. If a circle is intersected by any edge then the centre of the circle is moved towards its

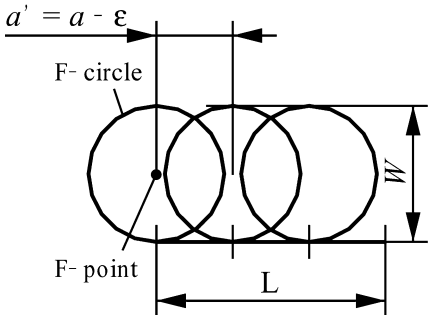

Fig. 3 Offset circles of a line segment

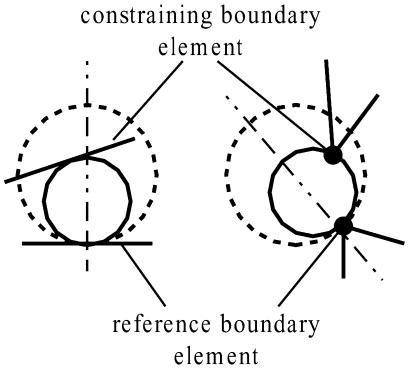

Fig. 4 An example for reference and constraining boundary elements

RBE and its radius is decreased until only concave vertices or intersecting edges touch the circles. Obviously, the tangential relationship with the RBE is maintained all along. At the end of this process, in the ideal case, there will be two boundary elements that touch the relevant offset circle; the RBE and a second one which is called the *constraining boundary element (CBE)*. Figure 4 shows two examples of the possible combinations: in the first case both the reference and the constrained boundary elements are line segments, in the second case both of them are concave vertices. If none of the offset circles have a CBE, that is, the offset distance is smaller than the radius of the smallest maximal inscribed circle, then the generated points (centres of offset circles) are on the internal offset curve of the cross-section.

Offset circle classification

The offset circles are arranged in a list and a type attribute is added to each circle depending on the relation between its diameter and the diameters of neighbouring circles. These are as follows: (Table 2):

1. Attribute *Z* (“zero”): A circle in which the radius degenerates to zero. This situation occurs when the RBE and the CBE of a circle are neighbouring edges of the cross-section that meet at a convex vertex.
2. Attribute *E* (“escape”): A circle with a diameter equal to the extrusion distance but preceeded by a sequence of circles with radius less than the extrusion width. That is, *E* circles do not have CBEs but the preceding circles do.

Table 2 Attributes of offset circles

	Circle Diameter	Next Diameter	Previous Diameter	Attr.
1	$=0$	>0	>0	Z
2	$=W$	$=W$	$<W$	E
3	$=W$	$<W$	$=W$	C
4	other			G

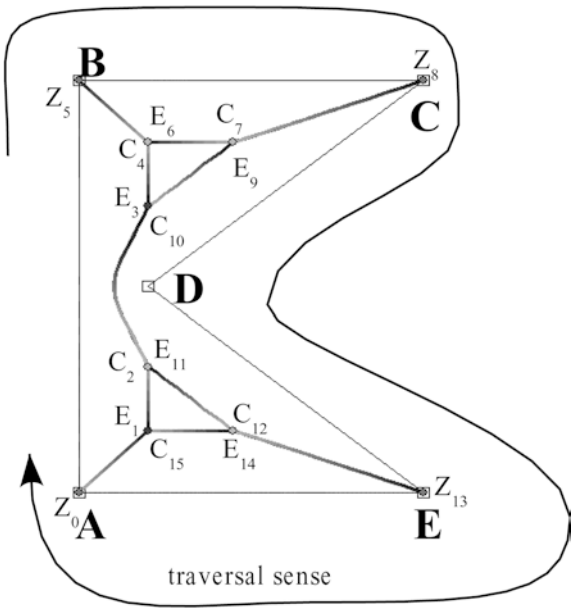

Fig. 5 An example case of offset circle attributes

3. Attribute *C* (“capture”): A circle with diameter equal to the extrusion distance but followed by a sequence of circles with radius less than the extrusion width. That is, *C* circles do not have CBEs but the following circles do.
4. Attribute *G* (“general”): A circle which is not a *Z*, *E* or *C* circle.

The centres of the circles listed above are named as *Z*, *E*, *C* and *G* points, respectively. The *Z*, *E* and *C* points are called *separator points* and are the centres of *separator circles*.

In order to clarify this classification, let us consider the example in Fig. 5, in which all the non- *G* points of a polygon offsetting are shown labelled by their attributes. Circles are not drawn here, but we reference them by their centres. The Z_0 circle has zero radius since its RBE is $\overline{AB}$ and its CBE is $\overline{EA}$ which meet at a convex vertex. Circles between Z_0 and E_1 have radii less that W ; E_1 is the first circle in the sequence which does not have any CBE. This latter condition holds for every circle between E_1 and C_2. C_2 is constrained by $\overline{DE}$; the next unconstrained circle is E_3. Between C_2 and E_3 vertex *D* will also appear as a CBE. After several unconstrained circles $\overline{BC}$ constrains C_4 and this condition remains for the rest of the circles of RBE $\overline{AB}$. The remaining circles are characterised in a similar way to Z_5 which is the zero-radius circle of $\overline{BC}$ constrained by $\overline{AB}$.

Based on the introduced classification it is possible to recognise special patterns in the list of separator points. Each pattern stands for a sub-sequence of points which correspond to the projection of particular surfaces of the MS to be calculated. Table 3 shows how these patterns are related to the sequence of projection points. (Note that the indices show the absolute position of the points

Table 3 Special patterns in the offset circle sequence

Pattern	Point sequence	Projection of
$\widehat{CZE}$	$C_i,G_{i+1},...,G_{i+n},Z_j,G_{j+1},...G_{j+m},E_k$	w-surface
$\widehat{CE}$	$C_i,G_{i+1},...,G_{i+n},E_j$	v-surface
$\widehat{EC}$	$R_i,G_{i+1},...G_{i+n},C_j$	h-surface

in the list so that $j=i+n+1$ and $k=j+m+1$.) (See also Fig. 1).

Locating junction circles

In order to facilitate the discussion of this topic we introduce two other, not explicitly registered offset circle attributes. Let circles immediately followed by an E circle be flagged with the G^E attribute and circles immediately preceded by a C circle are flagged with the G^C attribute (Fig. 6).

It can be seen that C and E points that are geometrically close to each other can be arranged in pairs such as (E_1,C_{15}), (C_2,E_{11}) etc. (Fig. 5). These pairs can be identified more precisely based on the fact that the RBEs and CBEs of the corresponding G^E and G^C circles permutate. These pairs make it possible to delimit domains of the so-called *junction circles, J circles* whose diameters are equal to W and are touched tangentially by the relevant RBEs and CBEs. This is a very important property, because J circles are the projections of the MISs of the extruded object which are constrained by four boundary elements of the object. J circles have the common properties that 1) their diameter is equal to W and 2) the distance between their centres and the closest boundary elements is equal to $W/2$. Such a circle occurs between a constrained and an unconstrained circle, because this is the domain where a circle conforming to these two conditions can be found. Subdomains between G^E and E circles or between C and G^C circles fulfill this requirement, where the closest boundary elements are given by the CBE of the G^C and G^E circles. The centre of the J circle sought is obtained by a numerical method which works for either edges or concave vertices of the cross-section. Instead of introducing a new circle, the foremost circle of the domain being searched is modified, that is, C and G^E circles are translated so that they touch the corresponding CBE (see Fig. 6) and the radius of circles of the latter type are set to $W/2$. We assign them the J^C and J^G attributes, respectively, indicating their original type.

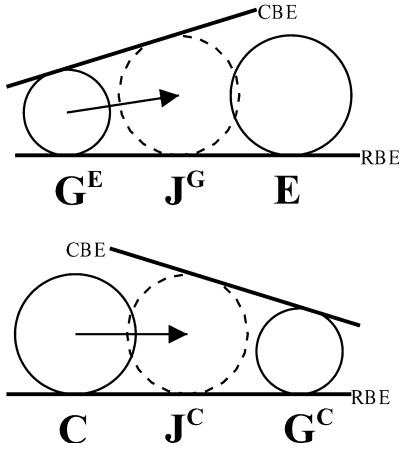

Fig. 6 Determiming the location of junction circles

Projection curve approximations

The task at this stage is to generate polylines composed of centres of offset circles that approximate the projection curves of medial surfaces. Before discussing this topic, we classify the possible approximate polylines in the following way:

- An approximate polyline of the projection of the bordering curve of an *h*-surface is called an *h-segment*.
- An approximate polyline of the projection of the bordering curve of a *v*-surface or a w^F-surface is called a *v-segment*.

The basis for the segment generation is the observation that the centres of circles of which RBE and CBE permutate lie on the same projection curve. Approximations of such curves contain implicit references to only two different boundary elements, and are called *A-B invariant segments* where A and B are the two referenced boundary elements. When traversing the list of offset circles it is not always necessary to start a new segment at every separator circle; instead, the algorithm checks every constrained circle to see whether it can be added to an existing segment. If so, the new circle will be appended to the list of circles of the existing segment. Unconstrained circles are not checked since *h* -surfaces do not have overlapping projection curves. The traversal starts with the first Z circle as in the previous case (Algorithm 1). We have chosen an order-dependent boundary traversal description for two reasons: (i) the explanation of the algorithms and implementations is simplified and (ii) distinguishing RBEs and CBEs is also a key element of the algorithm since there can be several offset circles that are touched by an RBE but are not constrained by any boundary elements, that is, no CBE (actually, these will form the boundary of h-surfaces).

```
Algorithm 1 Medial surface approximation
Require: circle c, segment s, segment list
    c:= first Z circle
  repeat
    if CBE(c) is not NULL then
      s:=invariant_segment(CBE(c),RBE(c))
      if s does not exist then
        if last_of() is not an empty v-segment then
          s:= new v-segment
          add s to
```

```
      end if
    end if
  else
    if last_of() is not an empty h-segment then
      s:= new v-segment
      add s to
    end if
  end if
  add the centre of c to s
until c is the first Z circle
```

The side-effect of the algorithm is that if there are no unconstrained offset circles in the model then only *v*-segments are created which approximate the 2D medial axis transform representation of the cross-section. This condition is fulfilled only if W is large enough; more precisely, for every P point inside the cross-section we can find a boundary element λ which is closer to that point than $W/2$, that is:

$$\forall P : P \rightarrow \left\{ \lambda | dist(P, \lambda) \leqslant \frac{W}{2} \right\}, P \in \Omega, \lambda \in \wedge$$

where Ω is the set of points of the cross-section and Λ is the set of boundary elements (edges and concave vertices) of the cross-section. Figure 7 shows an example of a generated 2D MAT.

At the end of the previous stage we have several segments that approximate MS curve projections. Generating surface approximations is done in two steps. Firstly, the distinct segments which approximate the bordering curves of the same MS surface have to be linked together; secondly, face sets have to be prepared that approximate the exact medial surfaces. The approximate face sets of surfaces are called *grids* and are distinguished by using the same prefix as their corresponding surfaces: *h-grid*, *v-grid*, etc.

v- and w^F-grids. Generation of these types of grids is quite simple, since both types can be generated from one distinct *v*-segment. The *v*-segment can be considered as the symmetry polyline, so the task is to "stretch" the segment vertically, that is, duplicate the segment points and translate the two instances towards the top and bottom faces. The amount of translation is equal to $W/2 - r_i$ where r_i is the radius of the constrained offset circle of which the centre is to be translated. This operation gives us two bordering segments of the grids; the other two bordering curves (vertical straight lines) can be obtained from the extremal points of the duplicated

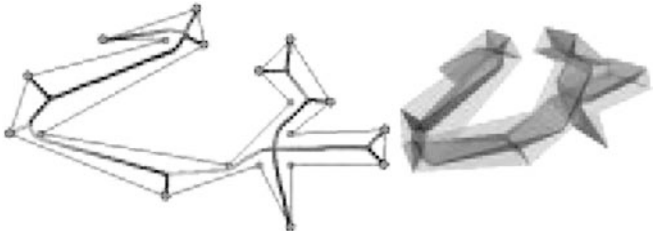

Fig. 7 The 2D MAT as the projection of *v*-surfaces

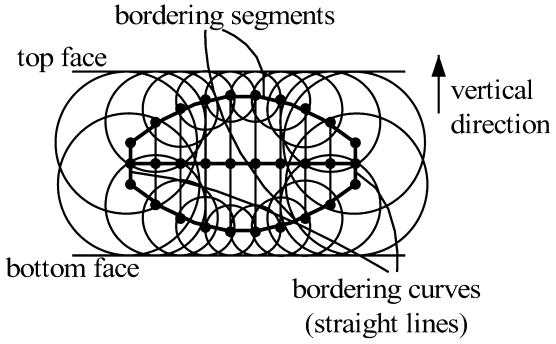

Fig. 8 The v-grid of the example in Fig. 1 from a viewing direction that is perpendicular to the A^T–F^T–F^B–A^B face

v-segment (Fig. 8). The algorithm creates one *v*-grid per *v*-segment as it traverses through the list of segments.

w^E-grids. Approximations of projections of w^E-curves are not represented explicitly in our model but only by two points. One of these points is always an F-point (see the section MS boundary points), while the other is located where the relevant F-circle touches its RBE. They are called *root-point* and *end-point*, respectively (Fig. 8). The root point might be located either on an *h*-segment or on a *v*-segment; therefore, it is worth generating w^E-grids after *h* - and *v*-grids since points of *v*-segments have already been duplicated. The end-point projections to the top and bottom faces create two grid points while the root point creates (in the case of a *v*-segment) two projection points on the corresponding bordering curves (Fig. 9). When the root point is on an *h*-segment then there is no need to project them and therefore we have three grid points altogether. After projecting the end-point to the top and bottom faces, we will have three or four grid points of the w^E-grid depending on whether the root-point was located on an *h*-segment or on a *v*-segment. Then, these points can be connected together to form the border of the w^E-grid. If a better resolution is needed (for example, the applications discussed in the section Applications) then the two non-vertical grid edges can be subdivided with respect to the pre-defined offset circle distance (see the section MS boundary points).

h-grids. The border of an *h*-grid is a loop composed of *h*-segments. The constituent segments of this loop do not follow each other continuously in the traversal

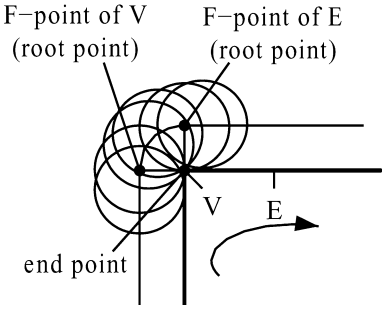

Fig. 9 Significant points of w^E-surfaces

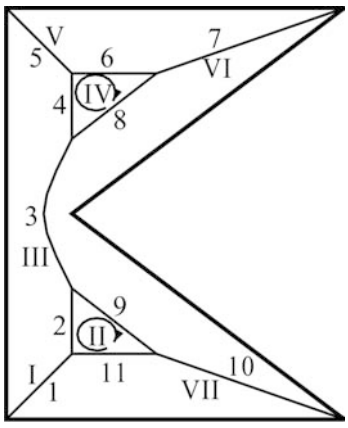

Fig. 10 An example of order of segments and grids

sequence of segments, therefore the matching endpoints have to be searched for. During the traversal the algorithm checks if the current *h*-segment has already been processed, if not, the segment will be registered as the starting segment of a new loop and a recursive subprocess will collect the corresponding segments of the current loop. Fig. 10 illustrates in which order *v* - and *h*-grids are created during traversal of the sequence of segments.

s-grids. As in the case of *v*-grids, *s*-grids can be created by using one single segment, which can be either an *h* - or *v*-segment. In the case of *h*-segments one of the bordering segments is the actual *h*-segment. The opposite bordering segment (which is coincident with a boundary edge of the extruded object) can be obtained by projecting points where the offset circles of the current bordering segment touch their RBEs onto the top face (s^T-grids) or the bottom face (s^B-grids). The other two bordering segments are formed by the extremal points of the discussed ones. The method is quite similar to that of *v*-grids except that we have to deal with the duplicated *v*-segments for which generation was discussed earlier.

The complexity of the algorithm

The number of boundary elements of the cross-section, the length of the cross-section edges and the preferred offset circle distance affect the number of generated offset circles, so this can be taken as the size of the problem (n). Assuming that there are no major changes regarding the overall dimensions of cross-sections in an application domain (therefore, there is no need to change the preferred offset circle distance), we can say that the problem size can be characterised by the number of cross-section elements.

Obviously, the offset circle generation and classification phases depend linearly on n. In the average case, the number of *J* circles grows linearly with n, that is, finding their exact location takes $O(n)$ time. During segment generation the list of existing segments is searched for each of the constrained offset circles, so considering the worst case, when the size of that list grows linearly with the problem size, the complexity of this phase is quadratic in n. In addition, the algorithm for sorting the offset circles increases the execution time linearly. The algorithm to create the MS approximations processes each segment once; therefore, its complexity is $O(n)$. Taking the most complex part of the present MS generation method, the overall complexity is $O(n^2)$. Table 4 summarises the above results.

Table 4 Complexity of sub-algorithms

Sub-algorithm	Complexity
Offset circle generation	O(n)
Constraining offset circles	$O(n^2)$
Offset circle classification	O(n)
Locating J circles	O(n)
Curve approximation	$O(n^2)$
Sorting offset circles	O(n)
Grid generation	O(n)

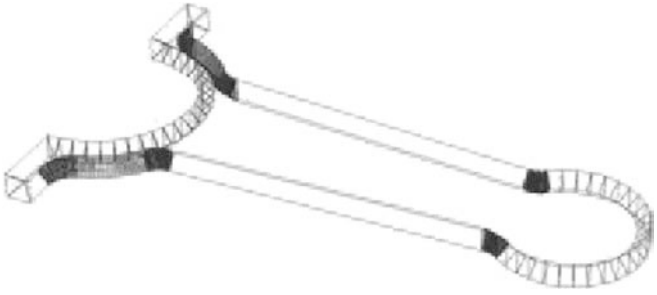

Fig. 11 An example case with curved cross-section elements

Test cases

The two major parameters of our MS creation method that can influence performance are the resolution of the initial cross-section (i.e., transforming the original curved sections to their polygon approximation) and the resolution of the MS to be created (a, see the section MS boundary points). The former depends on the number of elements (edges and concave vertices) in the cross-section while the latter is driven by the user-defined preferred distance of offset circles. We present two example cases here in order to examine these aspects by means of them.

In the first example the cross-section contains curved edges approximated by polylines (Fig. 11). The measurement of performance is done by applying different resolution factors (R) to these arcs using the following parameters:

- The lengthwise overall dimension of the example part was 230 mm.
- The preferred initial offset circle distance was set to 1 mm.
- The extrusion width (W) was set to 10 mm.

Table 5 Test results of the analysis of the first example case

R	n_E	n_C	$\bar{a}$ [mm]	n_S		T [sec]	
				top.	geo.	top.	geo.
8	147	1225	7.75	233	148	0.17	0.17
16	283	1817	6.30	425	283	0.44	0.49
24	419	2417	5.53	617	419	0.93	0.93
32	555	3097	4.93	809	508	1.65	1.60
40	691	3777	4.55	1001	634	2.47	2.41

Legend: R: resolution, n_E: number of cross-section elements (line segments and concave vertices), n_C: number of offset circles, $\bar{a}$: average distance of centres of offset circles, n_s: number of surfaces, T: execution time (including normal overhead)

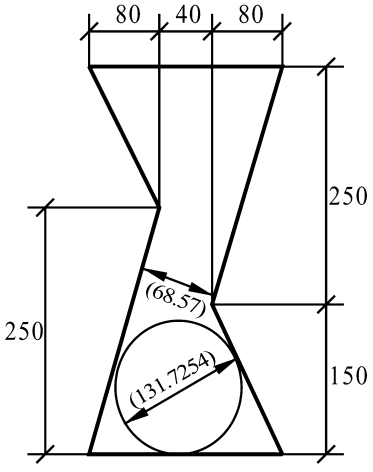

Fig. 12 A second example case

The results are summarised in Table 5[1]. The example is also useful to illustrate producing a minimum number of offset circles per RBE; this is the reason for the decreasing tendency of the average distance of offset circle centres. Since for bigger R we will have more line segments with smaller length the total number of offset circles is increasing.

We also used this example to examine the behaviour of the Renner-Stroud algorithm and we arrived at the following conclusions. Firstly, as mentioned in the Introduction, the generated MS is approximated by straight lines; however, the MS points are exact. Secondly, the algorithm has problems when dealing with MS points that are equidistant from more than four boundary elements (the algorithm generates multiple identical points causing increased complexity). This is the case, in the example object, at the two circular branches where W is equal to the difference of the external and internal radii. Thirdly, the MS generation took almost one and a half hours on a HP 9000 workstation (as opposed to a few seconds at most for our method according to Table 5), although it ran under the test harness module of the ACIS solid modeller.

The second example measures how calculation times depend on the resolution of the MS, that is, the predefined preferred distance of offset circles (Fig. 12); five preferred offset circle distance values (0.1, 0.08, 0.06, 0.04 and 0.02 mm) were combined with 12 extrusion width values (12, 32... 232 mm). The latter values were chosen so that either *v*-surfaces or *h*-surfaces were not generated at the extremal values. Fig. 13 shows MSs of the second example case. Fig. 14 shows graphically the resulted calculation times for each combination of the values. Fig. 15 shows the generated surfaces for $W = 32$, 132 and 232 mm (for clarity, only *h*-, *v*- and w^F-surfaces are shown). (Fig. 16 shows the layout of a vertical hoisting device).

The preferred offset circle distance (a) has a great impact on the precision of the result. Let us consider the degenerate case in Fig. 15 where a regular hexagon with

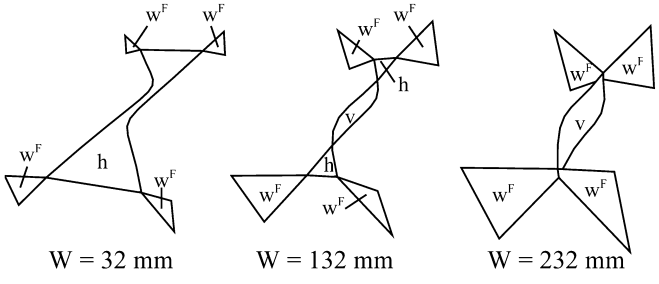

Fig. 13 MSs of the second example case

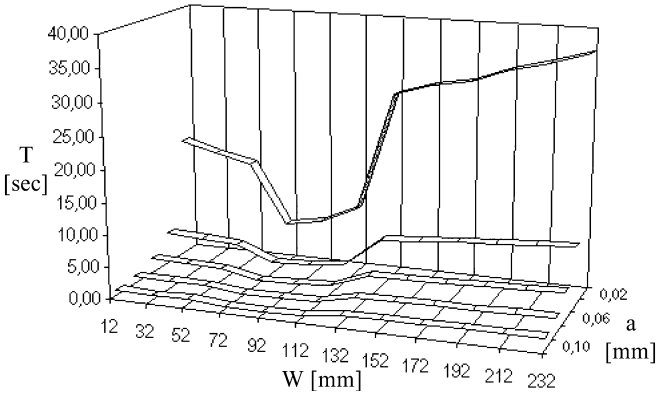

Fig. 14 Computation times as functions of extrusion distance and a resolution of the MS

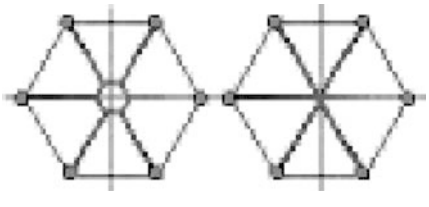

Fig. 15 The MS of a hexagonal bar with different preferred offset circle distance values

[1]The execution time was measured on a 400 MHz PC.

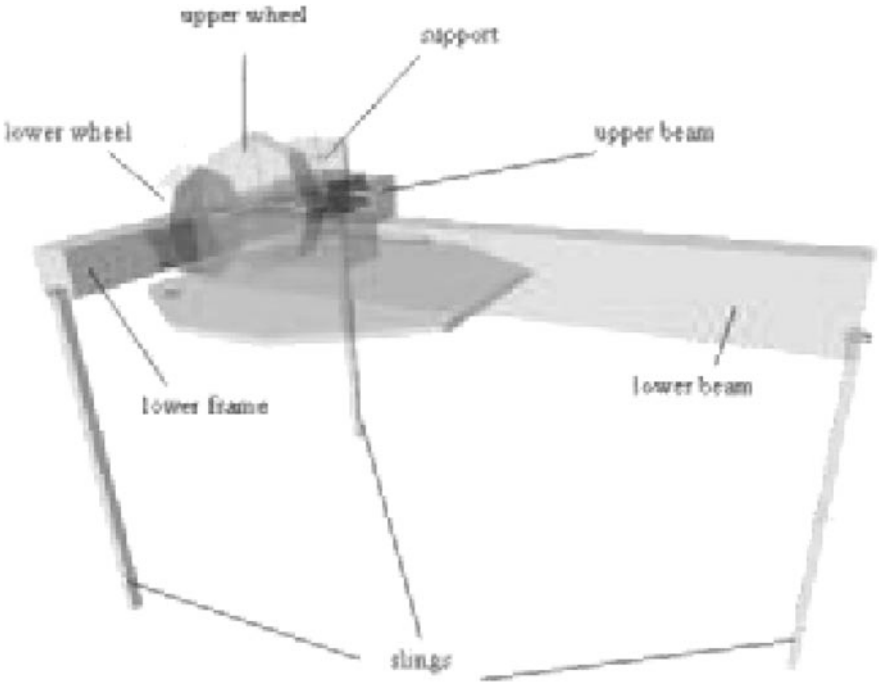

Fig. 16 The layout of a vertical hoisting device. (APCO Technologies SA)

50 mm edges was extruded by 300 mm. The left MS was created with a 10 mm preferred offset circle distance while this value was 5 mm in the second case. Our experiments showed that the algorithm works with acceptable precision if the preferred offset circle distance is one tenth of the shortest internal offset curve of the cross-section, that is:

$$a = \min\left(\frac{L_{\min}}{10}, \frac{\frac{w}{2}(\pi - \alpha_{\max})}{2}\right)$$

where L_{min} is the shortest cross-section edge and α_{max} is the angle of two edges that emanate from the bluntest concave vertex.

Other approaches may have serious problems with the abovementioned problem. For example, the original Reddy-Turkiyyah [11] and Renner-Stroud [14] algorithms would traverse the lengthwise axis of the object several times, because of the redundant internal sphere constraining elements. However, the latter approach has an improved version that overcomes this problem; in this case overconstrained boundary elements, are identified and counted with only once during the medial surface calculation.

Applications

As the Introduction summarises, medial surface representations have many uses in engineering tasks. However, this section focuses on a relatively new area of their application, namely the spatial layout design of mechanical assemblies [3, 2]. A Swiss industrial project whose goal was to establish the modelling principles of such layouts was successfully concluded with the collaboration of the CAD/CAM laboratory of the Swiss Federal Institute of Technology in Lausanne (EPFL). The future work of this research is to find tools to do the geometric evaluation of such layouts effectively; using the medial surface representation is a promising direction for this purpose.

In our modelling approach, the layout of a mechanical assembly is a skeleton where the components are represented by placeholders for the final physical assembly components, that is, parts or rigid sub-assemblies. These placeholders are called *design spaces*.

As soon as the layout is defined, it has to be checked against some design requirements in order to prove its correctness (functional verification, collision detection, etc.).

Ordinary interference detection methods work based on the fact that a point in space cannot be occupied by two different components at the same time. However, in our layout design methodology designers work with fuzzy geometries (design spaces), so this restriction cannot be applied as is.

Detecting the interference between two facetted objects by means of existing approaches is an easy task. However, if we want to find characteristic directions in the interference area, the face set representation has to be replaced by another one which is capable of returning such information. One possibility is to approximate the objects with internal spheres, since the intersection of two spheres can always be characterised by a vector which is defined by the centres of the two spheres and the interference volume (V), that is:

$$V = \int_{r=L}^{R_1} \left(R_1^2 - r^2\right)\pi dr + \int_{r=L_2}^{R_2} \left(R_2^2 - r^2\right)$$

$$L_1 = \frac{R_1^2 - R_2^2 + L^2}{2L} \quad L_2 = \frac{R_2^2 - R_1^2 + L^2}{2L}$$

where L is the distance of the centres of the two spheres, R_1 and R_2 are their radii. This interference vector can be interpreted as a repulsive force vector between two points embodied by the centres of the actual spheres. Summing up all the sphere-sphere intersection vectors for two given bodies gives a resultant repulsive force vector and a moment with respect to the centres of the objects. By means of the iterative application of such vectors we arrive to a configuration where the interferences between design spaces is minimal. Then the interfering portions of the design spaces are stored into the layout model for further processing in later design stages.

Conclusions

In this paper we have presented a method for calculating the approximate medial surface of objects created by the perpendicular sweeping of a planar cross-section. For this type of object the algorithm is stable, fast and robust and produces a fair approximation. We could summarise its main characteristics in the following way:

- *Robustness:* The algorithm does not involve heavy numerical computations, but fast linear algebra based calculations. Moreover, inaccuracies are always maintained and computational errors are corrected during operation.
- *Exactness:* If a relatively small MS resolution a is specified by the user, then a fairly exact approximation is created in a short time. In this case curved MS surfaces are approximated with a dense set of planar faces.
- *Speed:* In the average case the algorithm is capable of generating acceptable results in milliseconds, so it can be used for "on-the-fly" calculations. Although real-time processing is not so practical in our problem domain, since usually the MS is generated only once at the beginning of a stage of an application.

Many of the existing methods give only a rough approximation of medial surfaces (e.g., generating straight lines instead of curves or producing wireframes instead of surface models). The other group of them give fairly exact results but imply a heavy computational load so it takes a long time to achieve acceptable results. Our method provides improvements in both aspects. However, it can be applied only to a restricted set of possible 3D objects.

The forthcoming stage of the development of this work will be the extension of the approach to non-perpendicular extrusions and non-linear extrusions (swept volumes). The authors also plan to extend the method so as to be able to handle cross-sections with curved boundaries.

Acknowledgements The authors wish to acknowledge the financial support of the Swiss CTI project 3DLM. We would also like to acknowledge project support from our project partners LIA/EPFL, EIVd, APCO and Precisionsoft. The authors also thank Dr. Ian Stroud for his valuable comments and contributions.

References

1. Blum H (1967) A transformation for extracting new descriptions of shape. In: Wathen-Dunn W (ed) Models for the perception of speech and visual form, pp 362–381, MIT Press, Cambridge, MA
2. Csabai A, Xirouchakis P (2001) Container spaces and functional features for top-down 3D layout design. In: Proceedings of the 21st Computers and Information in Engineering Conference, Pittsburgh, PA, September 2001
3. Csabai A, Xirouchakis P, Taiber J (1998) Geometric constraint solving and applications. Springer, Berlin Heidelberg New York
4. Dutta D, Hoffmann CM (1993) On the skeleton of simple CSG objects. J Mech Des 115:87–94
5. Etzion M, Rappaport A (1999) Computing the voronoi diagram of a 3-D polyhedron by separate computation of its symbolic and geometric parts. In: Proceedings of the 5th ACM Symposium on Solid Modeling and Applications, Ann Arbor, MI, June 1999
6. Gürsoy HN, Patrikalakis NM (1991) Automated interrogation and adaptive subdivision of shape using medial axis transform. Adv Eng Soft 13(5–6):287–302
7. Gürsoy HN, Patrikalakis NM (1992) Automatic coarse and fine surface mesh generation scheme based on medial axis transform: Part I: algorithms. Eng Comp 8(3):121–137
8. Gürsoy HN, Patrikalakis NM (1992) Automatic coarse and fine surface mesh generation scheme based on medial axis transform: Part II: implementation. Eng Comp 8(4):179–196
9. Held M, Lukács G, Andor L (1994) Pocket machining based on contour-parallel tool paths generated by means of proximity maps. Comp Des 26(3):189–203
10. Lee Y-G, Lee K (1997) Computing the medial surface of a 3-D boundary representation model. Adv Eng Soft 28:593–605
11. Reddy RM, Turkiyyah M (1995) Computation of 3D skeletons using a generalized delaunay triangulation technique. Comp Des 29(9):677–694
12. Renner G (2001) Computation of medial surfaces. Computer and Automation Research Institute, Hungarian Academy of Sciences. Technical manual
13. Renner G, Stroud I (1997) Medial axis transform skeletonization and feature extraction. In: The World Congress on Intelligent Manufacturing Systems, Budapest, Hungary, June 1997
14. Renner G, Stroud I (1997) Medial surface generation and refinement. In: Pratt, Sriram, Wozny (eds) Product modelling for computer integrated design and manufacture, Chapman & Hall, London
15. Sheehy DJ, Armstrong CG, Robinson DJ (1996) The mathematics of surfaces VI. Oxford University Press, Oxford, UK
16. Sheehy DJ, Armstrong CG, Robinson DJ (1996) Computing the medial surface of a solid from a domain delaunay triangulation. In: Proceedings of the Third ACM Symposium on Solid Modeling and Applications, Salt Lake City, UT, May 1996
17. Sherbrooke EC, Patrikalakis NM, Brisson E (1996) Computation of the medial axis transform of 3-D polyhedra. In: Proceedings of the 3rd Symposium on Solid Modeling and Applications, Salt Lake City, UT, May 1996
18. Sherbrooke EC, Patrikalakis NM, Brisson E (1996) An algorithm for the medial axis transform of 3D polyhedral solids. Transactions on Visualization and Computer Graphics, 2(1):44–61, March 1996.
19. Storti DW, Turkiyyah GM, Ganter MA, Lim CT, Stal DM (1997) Skeleton-based modeling operations on solids. In: Proceedings of the Symposium on Solid Modeling and Applications, ACM Press, New York